

AGDD **The detector description framework**

- **Goal**
- **The design pattern**
- **The architecture**
- **The generic model**
- **Positioning volumes**
- **Graphical tools**
- **The C++ framework**

**Christian Arnault, Stan Bentvelsen, Steven Goldfarb,
Julius Hrivnac, Jean-François Laporte, Christopher Lester,
RD Schaffer, Marc Virchaux**

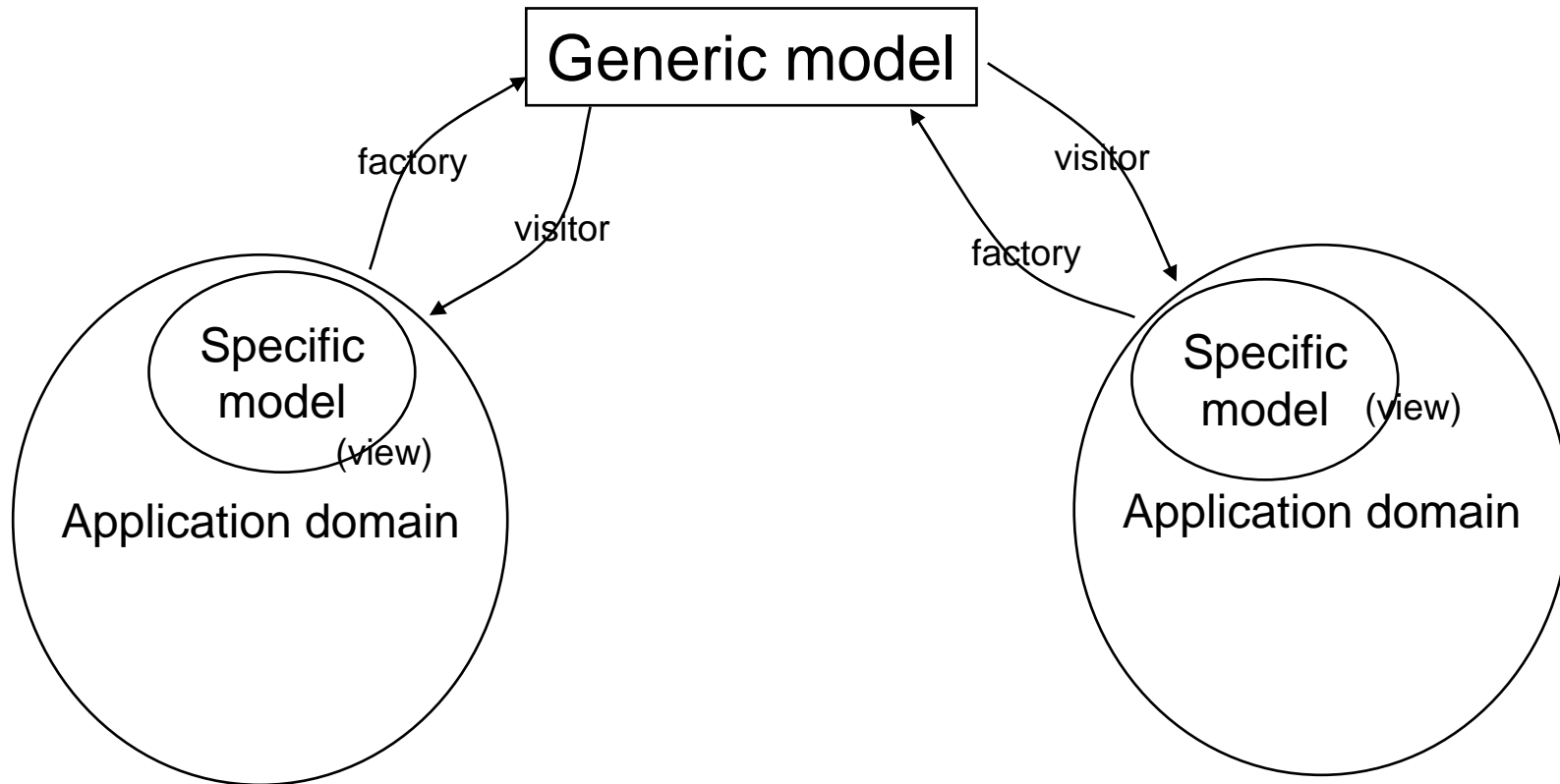
AGDD The detector description framework

- **Goals**

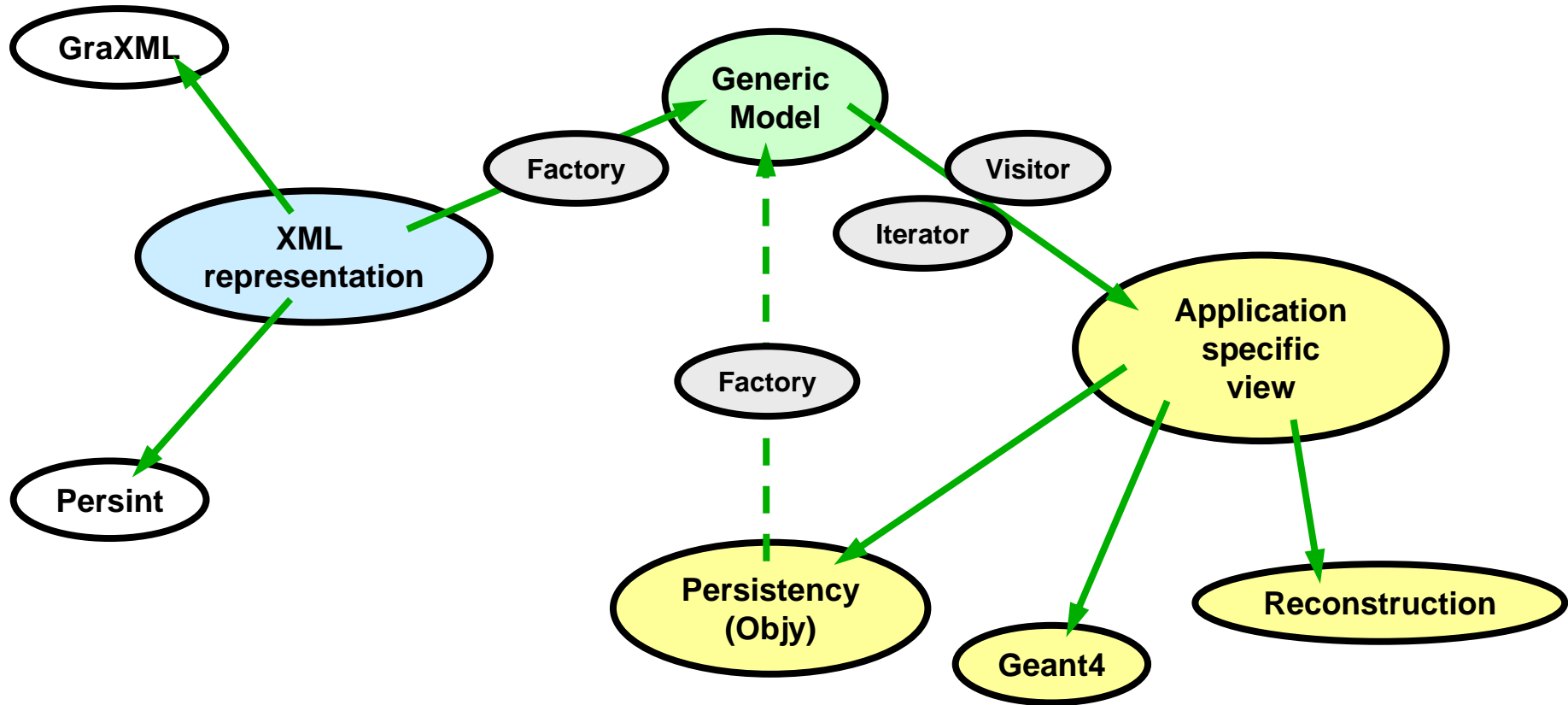
- provide a generic and common framework for describing all detector description information, independently of specific applications
- should not be dependent on specific constraints (such as Geant optimisations, reconstruction algorithms, ...)
- should not be limited to geometrical information
 - » logical organisation of the detector
 - » naming scheme(s)
 - » multiple views (structural, read-out, trigger, ...)
 - » physical properties of the detectors
- should support simple relationships with alignment data
- simple and portable implementation



AGDD The design pattern



AGDD The architecture



AGDD The Geometry Database

Inspired by Geant4:

1 Creation of Volumes:

⇒ **generic and descriptive**

– Elementary solids

Basic building blocks
characterized by shape,
dimension and material

- box
- tube
- trapezoid ...

– Boolean solids

boolean operations on 2 or
more volumes

- union
- intersection ...

– Logical grouping of volumes

- composition
- stack

2 Positioning of Volumes:

⇒ **description of the instantiation of the volumes**

– Absolute or relative, single or multiple positioners

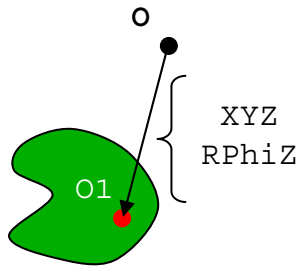
Define how the coordinate origin and the transformation matrix of a composed volume is computed from the origin of the constituents, through parameterized algorithms

- posXYZ
- posRZPhi
- mposPhi
- mposZ
- axisPos
- axisMPos

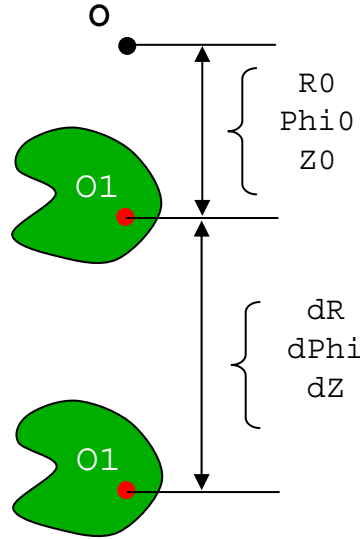


AGDD Positioning volumes

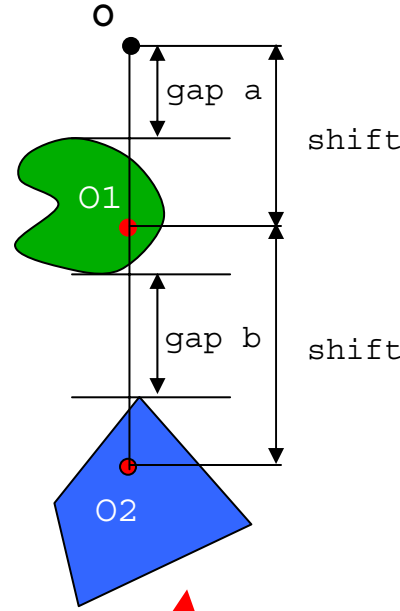
Single Absolute Positioners



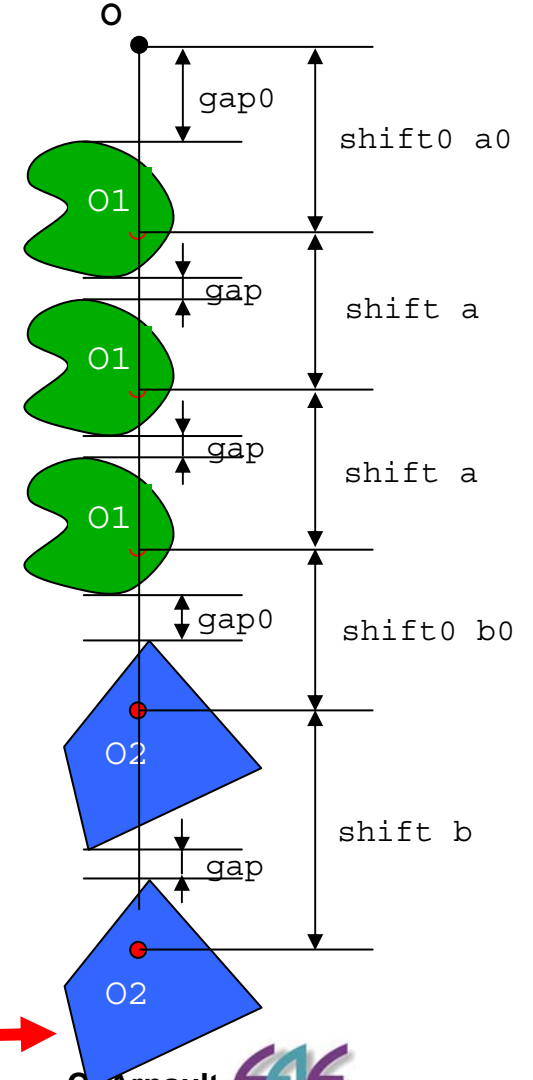
Multiple Absolute Positioners



Single Relative Positioners



Multiple Relative Positioners



```
<composition ...>
  <posXYZ volume=green ...>
</composition>

<composition ...>
  <mposZ volume=green ...>
</composition>
```

```
<stackZ ...>
  <axisPos volume=green gap=a>
  <axisPos volume=blue gap=b>
</stackZ>
```

```
<stackZ ...>
  <axisMPos volume=green ncopy=3 shift0=a0 shift=a>
  <axisMPos volume=blue ncopy=2 shift0=b0 shift=b>
</stackZ>
```



AGDD Simple example XML implementation

```
<?xml version="1.0"?>
<!DOCTYPE AGDD SYSTEM "AGDD_1.04.dtd">

<AGDD DTD_version="v4">

<!-- Workshop example file
*****
-->
<section name      = "WK"
      version      = "1.0"
      date         = "Wed Oct 27"
      author       = "Detector Description group"
      top_volume   = "ATLAS"
      DTD_version= "v4"      >

<tubs name="WK_tube" material="Aluminum" Rio_Z="0. 15. 1000." />

<composition name="WK_layer" >
  <mposZ volume="WK_tube" ncopy="10" dZ="30." rot="0 90 0" />
</composition>

<composition name="WK_multilayer" >
  <posXYZ volume="WK_layer" X_Y_Z=" 0 0 0" />
  <posXYZ volume="WK_layer" X_Y_Z=" 0 30 0" />
  <posXYZ volume="WK_layer" X_Y_Z=" 0 60 0" />
  <posXYZ volume="WK_layer" X_Y_Z=" 0 90 0" />
</composition>

<composition name="ATLAS" >
  <posXYZ volume="WK_multilayer" />
</composition>

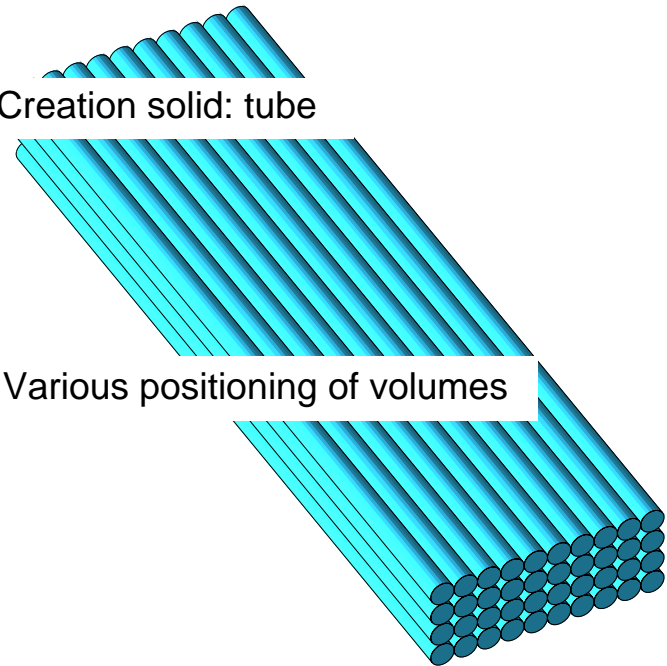
</section>
</AGDD>
```

AGDD Header, version of DTD

Section: sub-detector + author + version

Creation solid: tube

Various positioning of volumes



AGDD The generic model

- The generic model
 - It is derived from the definitions installed in the AGDD.dtd file
 - Since XML is *not* object-oriented, some *tuning* was needed (eg. To supply inheritance relationships)
 - Naming convention combines the original naming (defined in the DTD) with the C++ naming conventions (prefixing with AGDD_, capitalising, member prefixing, etc...)

```
AGDD
AGDD_Section
AGDD_Units
```

```
AGDD_Volume
    AGDD_Solid
    AGDD_Composition
    AGDD_Union
    AGDD_Intersection
    AGDD_Subtraction
    AGDD_Stack
```

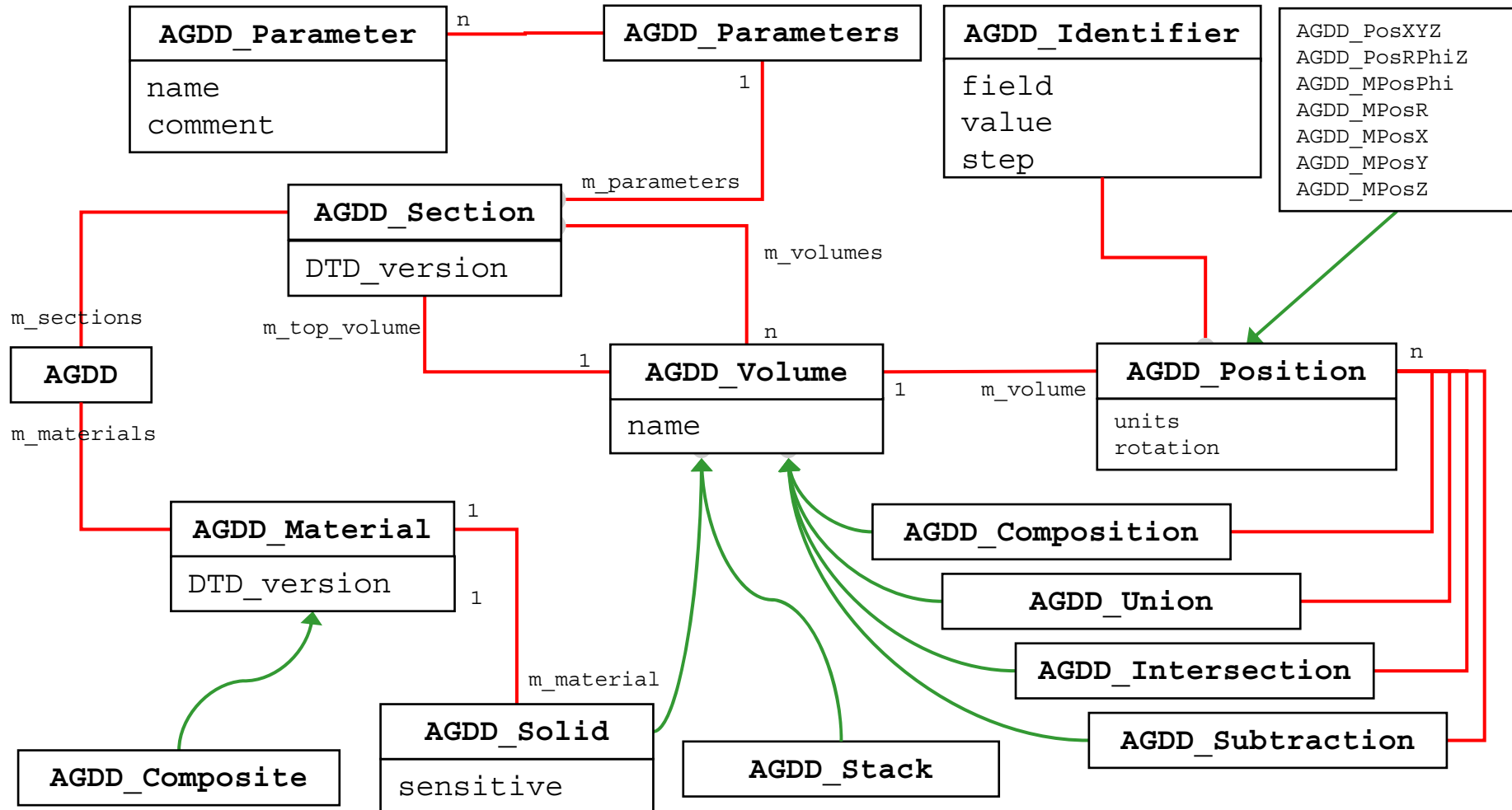
```
AGDD_Solid
    AGDD_Box
    AGDD_Trapezoid
    AGDD_Tubes
    AGDD_Cons
```

```
AGDD_Position
    AGDD_PosXYZ
    AGDD_PosRPhiZ
    AGDD_MPosPhi
    AGDD_MPosR
    AGDD_MPosZ
```

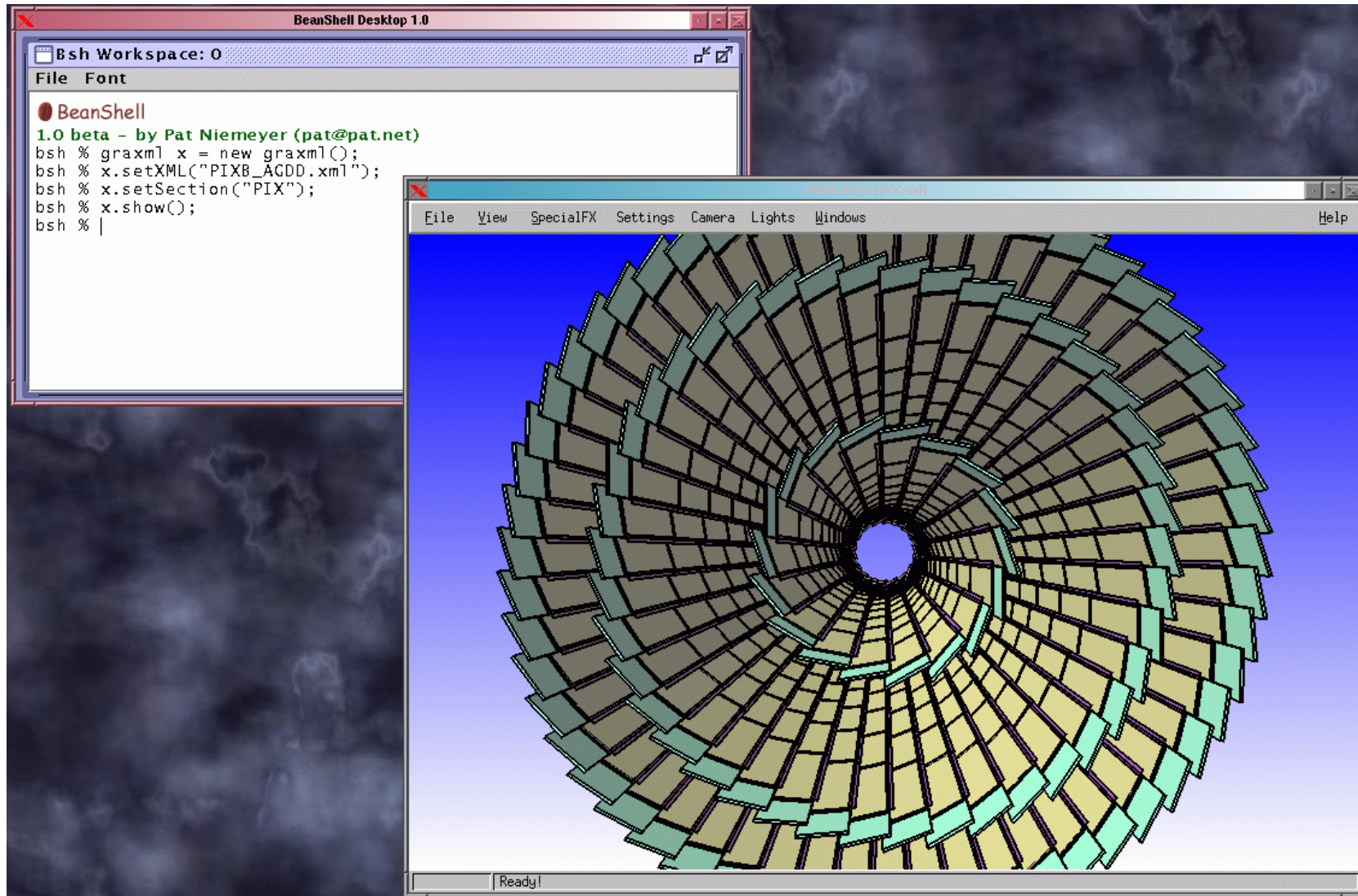
```
AGDD_Material
    AGDD_Element
    AGDD_Composite
AGDD_AddMaterial
```



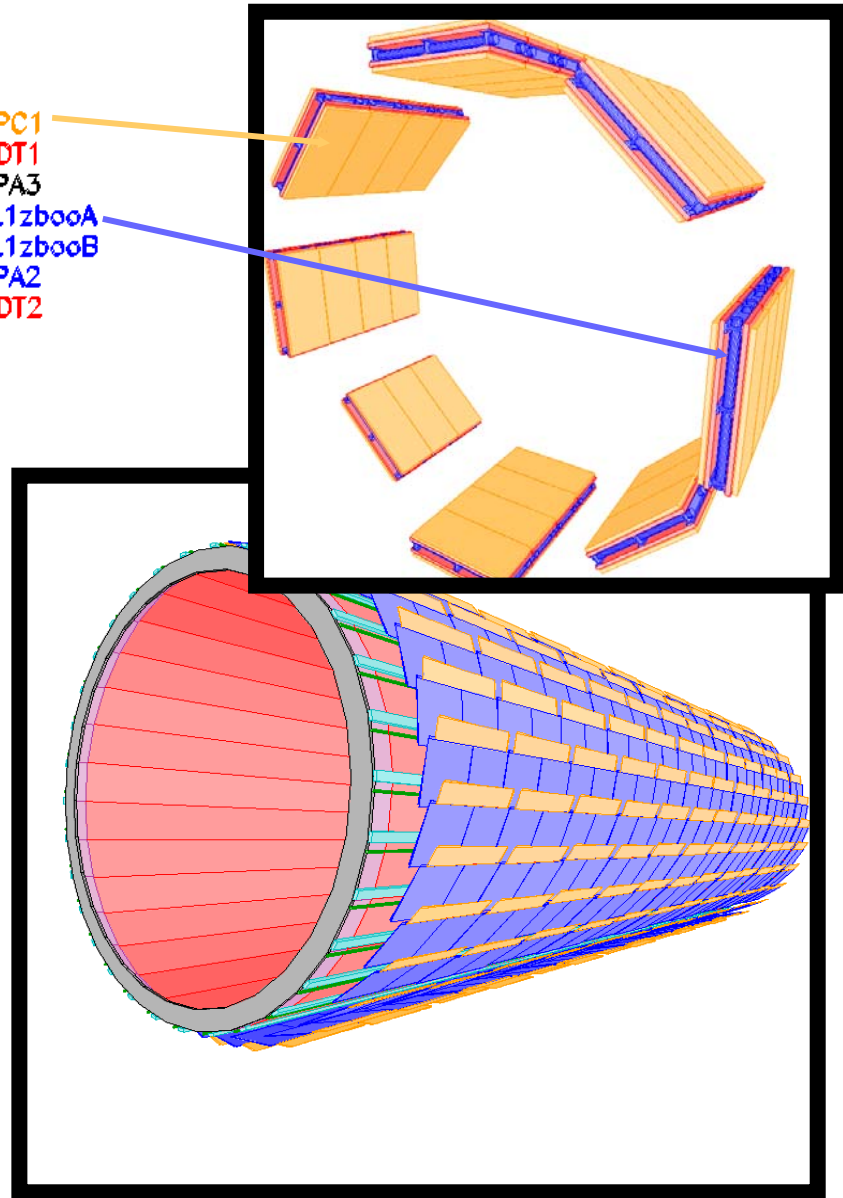
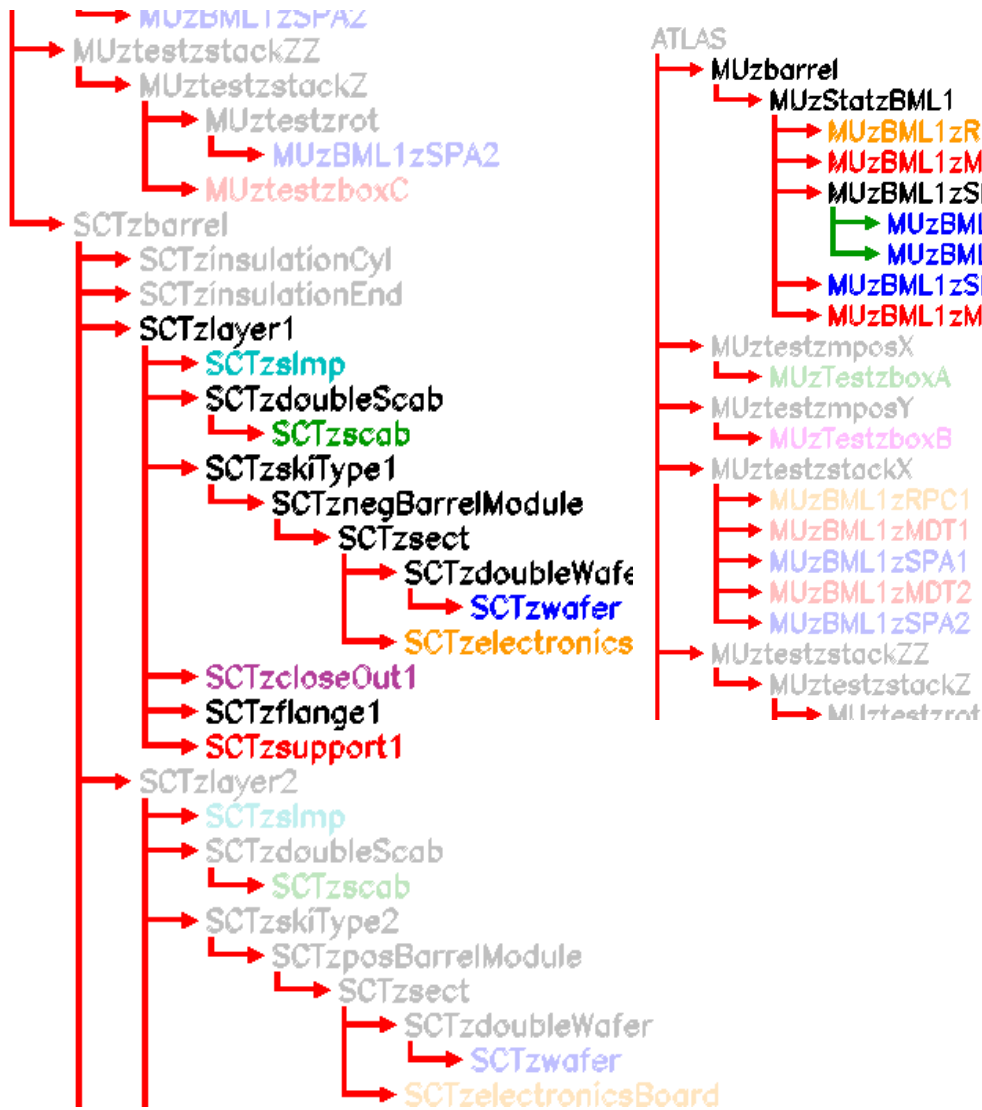
AGDD The generic model



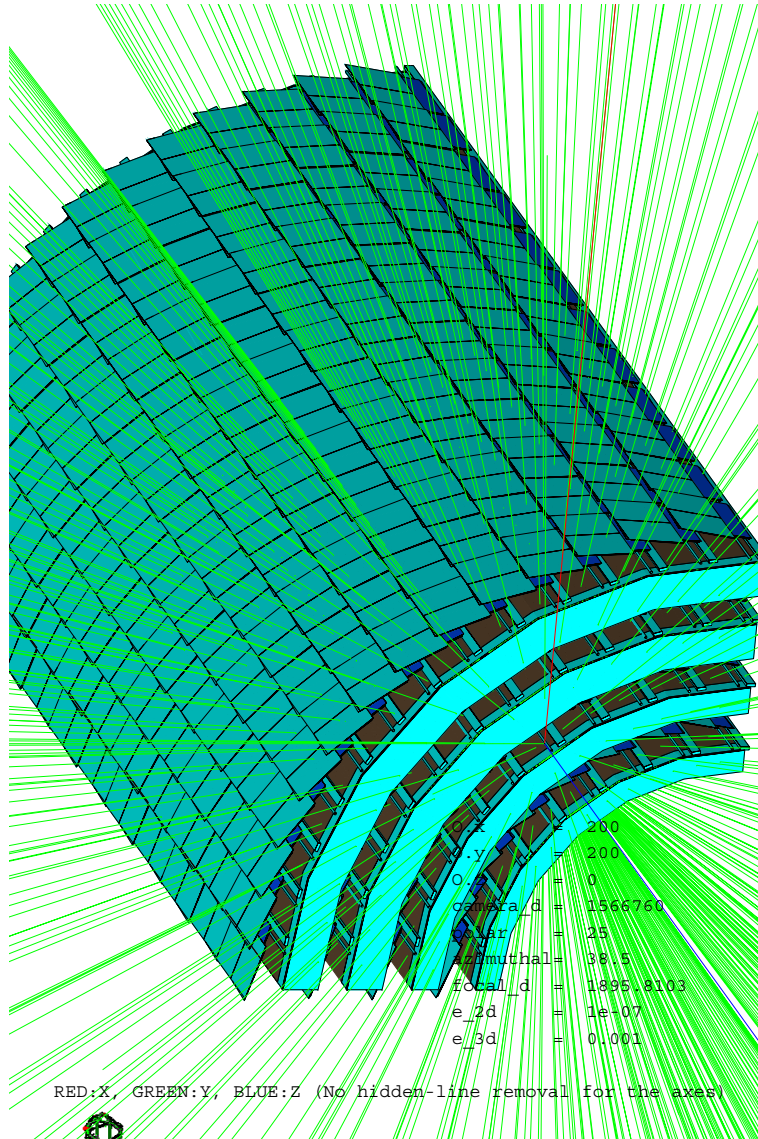
AGDD Graphical tools : GraXML (Java)



AGDD Graphical tools : Persint



AGDD G4Builder: interface to GEANT4



RED:X, GREEN:Y, BLUE:Z (No hidden-line removal for the axes)



ATLAS April 2000

12

- Set of classes, as client to the generic model, to transform the AGDD description to Geant4 geometry
 - Make use of the Material database in AGDD
- Generic, i.e. Detector independent.
 - Price paid in speed
- Track particles through geometry
 - **Example:** Scan geantinos in (η, ϕ) to determine characteristics of the ATLAS SCT geometry

S. Bentvelsen

C. Arnault



AGDD The DTD

```
<!ENTITY % DTD_constraint 'DTD_version ( v4 ) #REQUIRED'>

<!ELEMENT AGDD ( materials | section )+>
<!ATTLIST AGDD %DTD_constraint;>

<!ELEMENT section ( box | trd | tubs | cons | union | intersection | subtraction |
composition | stackX | stackY | stackZ | parameters )+>
<!ATTLIST section name CDATA #REQUIRED top_volume IDREF #REQUIRED %DTD_constraint;>

<!ENTITY % units 'unit_length ( mm | m ) "mm" unit_angle ( deg | mrad ) "deg"'>

<!ELEMENT identifier EMPTY>
<!ATTLIST identifier field CDATA #REQUIRED value CDATA "0" step CDATA "0">

<!ENTITY % solid_properties '%volume_properties;
material IDREF #REQUIRED
sensitive ( true | false ) "false"'>

<!ELEMENT box EMPTY >
<!ATTLIST box X_Y_Z CDATA #REQUIRED %solid_properties; %units;>

<!ENTITY % any_position ' posXYZ | posRPhiZ | mposR | mposPhi | mposX | mposY | mposZ '>

<!ELEMENT composition ( %any_position; )+ >
<!ATTLIST composition %volume_properties; envelope IDREF #IMPLIED>

<!ENTITY % any_relative_position ' axisPos | axisMPos '>

<!ELEMENT stackX ( %any_relative_position; )+ >
<!ATTLIST stackX %volume_properties;>

<!ENTITY % position_properties 'volume IDREF #REQUIRED rot CDATA "0 0 0" %units; '>

<!ELEMENT posXYZ ( %any_identifier; )* >
<!ATTLIST posXYZ X_Y_Z CDATA "0 0 0" %position_properties;>
```

DTD keyword
element or entity
attribute
value
reference



AGDD The C++ framework

```
AGDD_Factory
AGDD_Factory& Expat_instance ();
AGDD_Factory& XML4C_instance ();

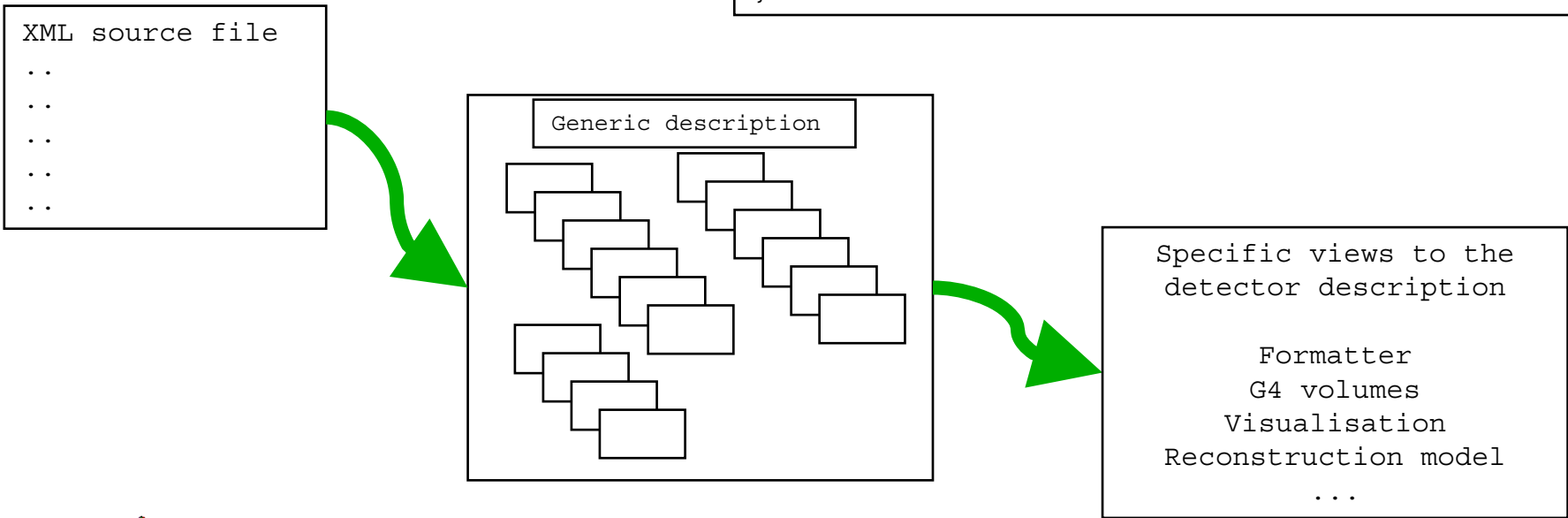
void build_materials (file_name);
void build_detector_description (file_name);

AGDD* get_detector_description ();

Void main ()
{
  G4_BuildMaterial material_builder;
  G4_BuildDetector detector_builder;

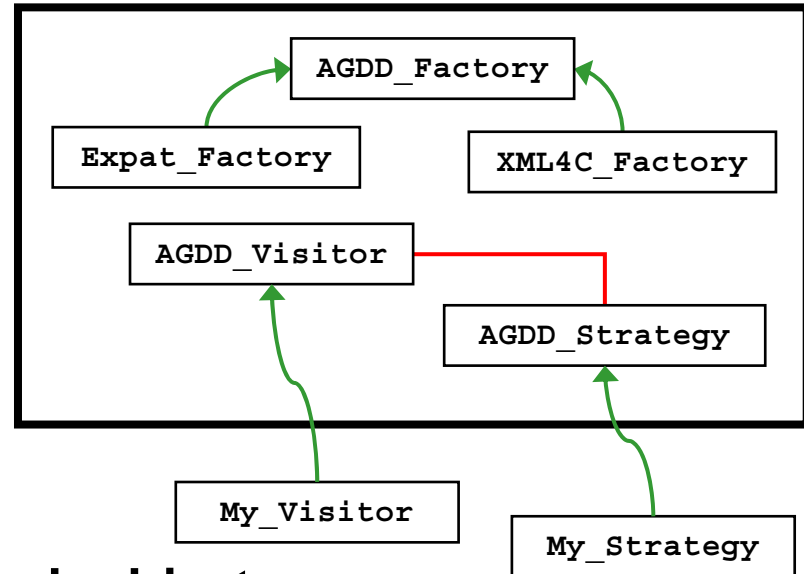
  AGDD* agdd = factory.get_detector_description ();

  material_builder.run (agdd);
  detector_builder.run (agdd);
}
```



AGDD The C++ framework

- **The Factory**
 - decodes the source file
 - instanciates objects
 - » sections, volumes, positions, ...
- **The visitor**
 - navigates the generic model
 - applies specific algorithms on each object
- **The strategy**
 - defines how to navigate the model



```
class G4_BuildMaterial : public AGDD_Visitor
{
    void visit_element (AGDD_Element* material);
    void visit_composite (AGDD_Composite* material);
};

class G4_BuildDetector : public AGDD_Visitor
{
    void visit_section (...) { std::cout << ... }
    void visit_solid (...);
};
```



AGDD The identification scheme

- **Logical identifiers**
 - numbering of tree-structured objects
 - independent from the specific design of the tree
 - ordered (can be used to key a map)
- **Ranges of identifiers**
 - specify a sub-set within a collection
- **Specialized collection**
 - contains identifiable objects
 - interfaced with ranges

```
Range r1 ("1/*/3-4");  
Range r2;  
  
r2.add (1);  
r2.add ();  
r2.add (3, 4);  
if (r2.match (a)) ...
```

```
Identifier a ("4/5/5");  
Identifier b;  
  
b << 1;  
b << 23;  
b << 4;  
int system_id = a[0];  
if (a < b) ...  
  
IdentifierMap detectors;  
  
detectors[a] = ecal;  
detectors[b] = muon;
```

```
MuonSpectrometer/StationName/eta/phi_sector/MDT/multilayer/tube_layer/tube  
MuonSpectrometer/StationName/eta/phi_sector/RPC/doublet/strip_layer_pair/strip_layer/strip  
MuonSpectrometer/StationName/eta/phi_sector/TGC/chamber_layer/chamber/strip
```



AGDD The identification scheme

Identifier
<pre> Identifier (); Identifier (const Identifier& other); Identifier (int numbers, ...); Identifier (const char* text); void add (int value); Identifier& operator << (int value); int& operator [] (int index); void clear (); int operator [] (int index) const; int fields () const; int operator == (const Identifier& other) const; int operator != (const Identifier& other) const; int operator < (const Identifier& other) const; int operator > (const Identifier& other) const; error_code last_error () const; operator string () const; </pre>

Range
<pre> Range (const char* text); Range (const Identifier& min, const Identifier& max); void add (); void add (int value); void add (int minimum, int maximum); void add_minimum (int minimum); void add_maximum (int maximum); int match (const Identifier& id) const; const field& operator [] (int index) const; int fields () const; Identifier minimum () const; Identifier maximum () const; identifier_factory factory_begin (); const_identifier_factory factory_begin () const; identifier_factory factory_end (); const_identifier_factory factory_end () const; operator string () const; </pre>

